Arduino

Co budeme potřebovat ?

1. Software :
   a) Arduino (IDE) - windows (64 b.), windows app, mac os,
      - linux (32 b., 64b., arm), online web
      https://www.arduino.cc

   b) UnoArduSim - windows (32b.)
      https://www.sites.google.com/site/unoardusim/

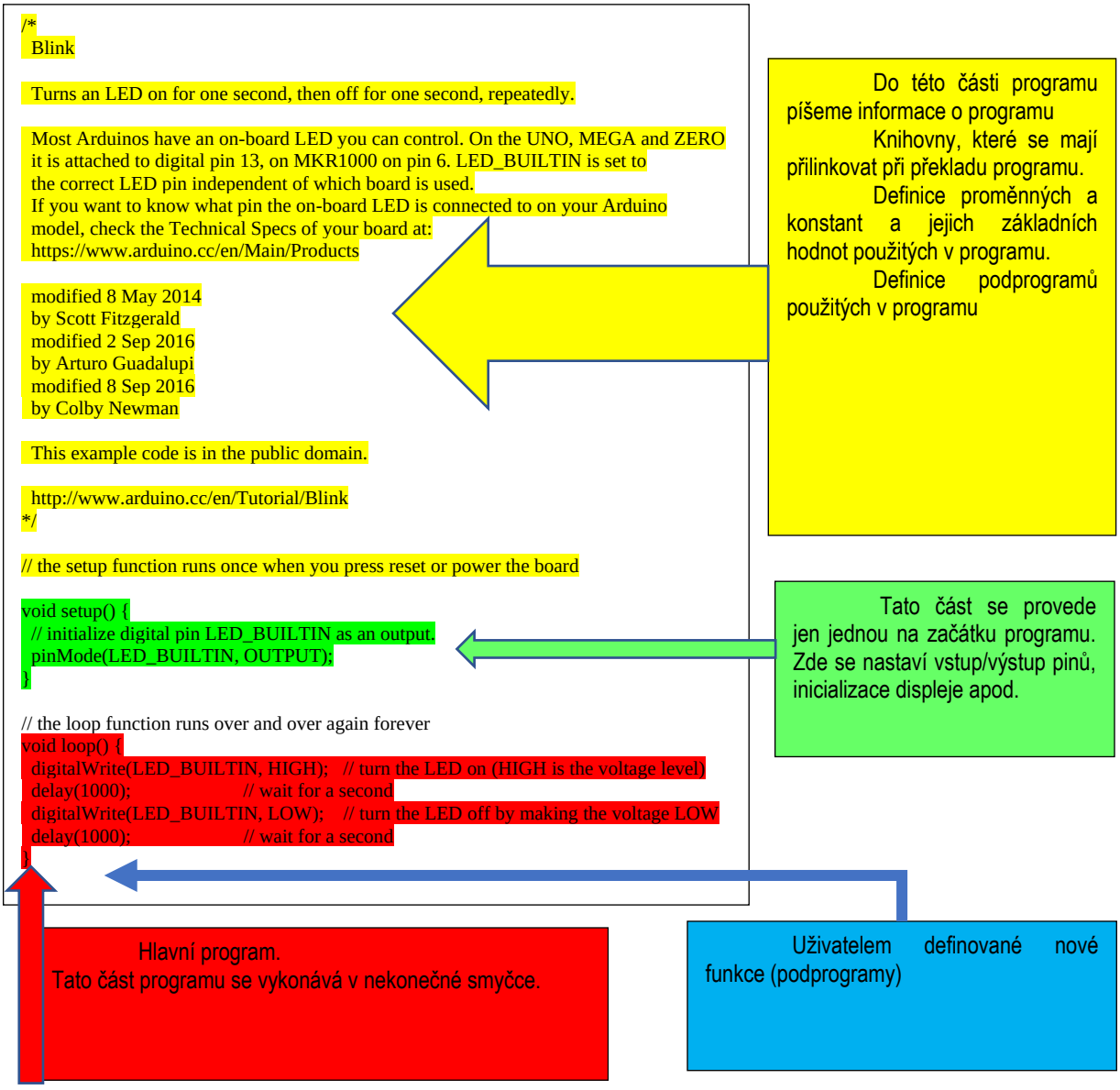
2. Hardware :
   a) Arduino Uno

   b) Propojovací USB kabel

   c) Auto + propojovací kabely

   d) PC pro programování, simulaci a nahrání

Struktura programu



```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the board

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Do této části programu píšeme informace o programu

Knihovny, které se mají přilinkovat při překladu programu.

Definice proměnných a konstant a jejich základních hodnot použitých v programu.

Definice podprogramů použitých v programu

Tato část se provede jen jednou na začátku programu. Zde se nastaví vstup/výstup pinů, inicializace displeje apod.

Hlavní program.
Tato část programu se vykonává v nekonečné smyčce.

Uživatelem definované nové funkce (podprogramy)

H můstek - Driver pro krokové a stejnosměrné pohony L298N

Driver je vhodný pro řízení krokových nebo stejnosměrných motorů. Řízení pohonu zajišťuje integrovaný H–můstek L298N. Je to vlastně jen zesilovač signálů. U stejnosměrných motorů lze řídit směr a rychlost otáčení.

Pro řízení se používají dva vodiče pro určení směru otáčení motoru. Můžete si přestavit jako byste připojili tyto dva vodiče na motor. Pokud na tyto vodiče přivedete stejnou logickou úroveň motor stojí. Pokud přivedete rozdílnou logickou úroveň motor se otáčí.

Funkce	motor - A	motor - B
STOP	0	0
Dopředu	1	0
Dozadu	0	1
STOP	1	1

Třetí vodič (řídící) slouží k regulaci výstupního výkonu motoru pomocí PWM regulace.
analogWrite(pin, 0-255) – 255 maximální výkon, kolem 150 začíná se motor točit (záleží na napájecím napětí pro motory).
digitalWrite(pin, 0 nebo 1) - Pokud ji nechceme využít pwm regulaci musíme na tento vývod přivést logickou jedničku. Při logické nule motor nedostane napětí na výstupu.

Pro roztočení motoru musí být splněny obě tyto podmínky:

1. Vstup motor-A a motor-B mají rozdílnou logickou úroveň
2. Řídící vodič má logickou 1 nebo přiveden PWM signál (1-255) – při hodnotách menších než 150 se motor většinou neroztočí.

Pro zastavení motoru musí být splněna alespoň jedna podmínka je jedno která:

1. Vstup IN1 a IN2 mají stejnou logickou úroveň
2. Řídící vodič má logickou 0 nebo přiveden PWM signál s hodnotou 0

Tyto vlastnosti můžeme dat do tabulky :

	D12 – motor1 A	D11 – motor1 B	D6 – motor 1 PWM
STOP	0	0	0
STOP	1	0	0
STOP	0	1	0
STOP	1	1	0
STOP	0	0	1
Dopředu	1	0	1
Dozadu	0	1	1
STOP	1	1	1

Úkol č.1 – Napište podprogram pro autíčko – Motor 1, dopředu – bez podprogramu

Napište program, který bude točit motorem 1 po směru jízdy. Využijte znalosti pro řízení výstupů. V simulátoru nenajdete tento typ motoru. Využijte signalizaci pomocí led diod.
Ovládání motoru je podle tabulky. Vývod PWM řídí rychlost otáčení motoru nastavte na logickou 1, vývod A-B řídí směr otáčení motoru. Motor1_A - High a Motor1_B - LOW.

```
#define motor1_a 12
#define motor1_b 11
#define motor1_pwm 6
#define motor2_a 8
#define motor2_b 7
#define motor2_pwm 5

void setup() {
  Serial.begin(9600);
  pinMode(motor1_a, OUTPUT);
  pinMode(motor1_b, OUTPUT);
  pinMode(motor1_pwm, OUTPUT);
}
void loop() {
  digitalWrite(motor1_a, HIGH);
  digitalWrite(motor1_b, LOW);
  digitalWrite(motor1_pwm, HIGH);
}
```

Po přenesení textu do programu arduino ide použij klávesovou zkratku pro autoformát Ctrl-t

	D12 – motor1 A	D11 – motor1 B	D6 – motor 1 PWM
STOP	0	0	0
STOP	1	0	0
STOP	0	1	0
STOP	1	1	0
STOP	0	0	1
Dopředu	1	0	1
Dozadu	0	1	1
STOP	1	1	1

Podprogram – uživatelsky definované funkce

Jsou funkce, které může uživatel vytvořit sám. Funkce je jakýsi soubor instrukcí „zabalený“ v jednom příkazu. Může mít vstupní parametry, se kterými dále pracuje. Obsahuje blok příkazů, které se při volání (spuštění) funkce provedou a také může vracet pouze jednu hodnotu. Zajímavé je, že každá funkce má určitý datový typ. Ten se liší podle typu dat, která vrací. Pokud funkce žádnou hodnotu nevrací, používá se speciální datový typ *void*. Důležité je nezapomenout na to, že proměnné definované v těle funkce není možné používat mimo tuto funkci. Ve funkci však lze používat proměnné definované na začátku programu. Funkce musí být definována mimo tělo jiných funkcí, nezáleží však, jestli je definovaná před, mezi nebo za funkcemi setup() a loop(). Většinou se však píší za smyčku loop.

```
//funkce může být tedy definována:
//tady
void setup() {
      //tady ne
}
//tady
void loop() {
      //tady ne
}
//tady
```

Již dříve jsme se setkávali s funkcemi, aniž byste to věděli. Například delay. Jako parametr používáme čas zpoždění.

Proč použít podprogram:

- Nemusí být celý program v sekci loop
- Možnost odladění části programů
- Lepší čitelnost programu
- Možnost použití stejné části programu ve více místech s různými parametry.

Možnost předání hodnot do podprogramu a z podprogramu:

Definice funkce

Aby funkce pracovala bez problému, potřebuje mít datový typ, název a závorky. U funkcí bez vstupních parametrů se kulaté závorky nechají prázdné (ale musí zde být).

```
int secti (int cislo1, int cislo2) {
      //vlastní program
      return cislo1+cislo2; //příkazem return předáme jednu hodnotu do hlavního programu
}
```

Typ proměnné, kterou bude podprogram vracet přes return. Typ void nevrací žádnou hodnotu do hlavního programu

Unikátní jméno podprogramu

Hodnoty přijaté z hlavního programu a přiřazení nových lokálních jmen pro proměnné. Platí jen v podprogramu. Pokud nepředáváme žádné hodnoty mezi závorky nic nepíšeme.

S parametry se pracuje stejně jako s proměnnými. Pokud funkce nějaké má, musíme je nadefinovat. Definice probíhá v kulatých závorkách. Pokud má funkce více parametrů, oddělují se čárkami.

V podprogramu můžeme použít i globální proměnné (ty co jsou definovány na začátku programu před sekcí setup. Tímto způsobem můžeme předat do podprogramu i vracet z podprogramu mnohem více proměnných. Pozor na záměnu lokálních a globálních proměnných.

```
int count;
void setup()
{
        count=0;
}

void loop()
{
        count=secti(count,1);
        delay(100);
}

int secti (int cislo1, int cislo2) {
        //program
        return cislo1+cislo2;
}
```

Funkce, které vrací hodnotu

Pokud má funkce něco vracet, musí mít jiný datový typ než void. Pro vrácení vybrané hodnoty se používá příkaz return. Pokud chceme vrátit řetězec znaků, nepoužívá se pole char[], ale datový typ String. Také není možné jednoduchým způsobem vrátit pole. Ostatní datové typy se používají stejně.

Nyní upravíme úkol 1 tím způsobem, že vše co bylo ve smyčce loop dáme do podprogramu a ten budeme volat v hlavní smyčce loop. Již předem jsme si odzkoušeli podprogram točení motoru dopředu pro další použití.

Úkol č.2 – Napište podprogram pro autíčko – Motor 1, dopředu – s použitím podprogramu

Napište program, který bude točit motorem 1 po směru jízdy. Využijte znalosti pro řízení výstupů. V simulátoru nenajdete tento typ motoru. Využijte signalizaci pomocí led diod.
Ovládání motoru je podle tabulky. Vývod PWM řídí rychlost otáčení motoru nastavte na logickou 1, vývod A-B řídí směr otáčení motoru. Motor1_A - High a Motor1_B - LOW.

```
#define motor1_a 12
#define motor1_b 11
#define motor1_pwm 6
#define motor2_a 8
#define motor2_b 7
#define motor2_pwm 5

void setup() {
  Serial.begin(9600);
  pinMode(motor1_a, OUTPUT);
  pinMode(motor1_b, OUTPUT);
  pinMode(motor1_pwm, OUTPUT);
}
void loop() {
  motor1_dopredu();
}

void motor1_dopredu(){
                  // zde dáte program pro točení motoru1 dopředu
}
```

Odzkoušejte funkci celého programu.

Nyní již umíte vytvářet podprogramy (funkce) a proto úpravou funkce motor1_dopredu vytvořte další funkce:

Úkol č.3 – Napište podprogram pro autíčko – Motor 1, dozadu – s použitím podprogramu

Úkol č.4 – Napište podprogram pro autíčko – Motor 1, stop – s použitím podprogramu

Úkol č.5 – Napište podprogram pro autíčko – Motor 1, odzkoušení funkcí

V hlavní smyčce volejte funkce v tomto pořadí

1. Motor1_dopředu
2. Čekej 200ms
3. Motor1_stop
4. Čekej 200ms
5. Motor1_dozadu
6. Čekej 200ms
7. Motor1_stop
8. Čekej 200ms

Motor1 by měl jet chvíli dopředu, zastavit, dozadu, zastavit a pořád dokola opakovat.

Nyní umíte ovládat otáčení motoru 1.

Stejné funkce vytvořte pro motor2

Úkol č.10 – Napište podprogram pro autíčko – Motor 2, dopředu – s použitím podprogramu

Napište program, který bude točit motorem 2 po směru jízdy. Využijte znalosti pro řízení výstupů. V simulátoru nenajdete tento typ motoru. Využijte signalizaci pomocí led diod.
Ovládání motoru je podle tabulky. Vývod PWM řídí rychlost otáčení motoru nastavte na logickou 1, vývod A-B řídí směr otáčení motoru. Motor2_A - High a Motor2_B - LOW.

	D8 – motor2 A	D7 – motor2 B	D5 – motor 2 PWM
STOP	0	0	0
STOP	1	0	0
STOP	0	1	0
STOP	1	1	0
STOP	0	0	1
Dopředu	1	0	1
Dozadu	0	1	1
STOP	1	1	1

Úkol č.11 – Napište podprogram pro autíčko – Motor 2, dozadu – s použitím podprogramu

Úkol č.12 – Napište podprogram pro autíčko – Motor 2, stop – s použitím podprogramu

Úkol č.13 – Napište podprogram pro autíčko

V hlavní smyčce volejte funkce v tomto pořadí

1. Motor1_dopředu + Motor2_dopředu
2. Čekej 200ms
3. Motor1_stop + Motor2_stop
4. Čekej 200ms
5. Motor1_dozadu + Motor2_dozadu
6. Čekej 200ms
7. Motor1_stop + Motor2_stop
8. Čekej 200ms

Autíčko by mělo jet chvíli dopředu, zastavit, dozadu, zastavit a pořád dokola opakovat.

Ukážeme jak předat do funkce nějaký parametr. V našem případě výkon pro motor1 dopředu.

Úkol č.15 – odzkoušej předání parametru do a z funkce

```
#define motor1_a 12
#define motor1_b 11
#define motor1_pwm 6
#define motor2_a 8
#define motor2_b 7
#define motor2_pwm 5

byte vykon, hodnota;

void setup() {
  Serial.begin(9600);
  pinMode(motor1_a, OUTPUT);
  pinMode(motor1_b, OUTPUT);
  pinMode(motor1_pwm, OUTPUT);
}
void loop() {
  for (vykon=0; vykon <= 100; vykon++) { //zkušební data pro výkon 0 -
100 %
    hodnota = motor1_dopredu(vykon);
    Serial.print(vykon);
    Serial.print(" - ");
    Serial.println(hodnota);
    delay(50);
  }
  delay(400);
}

int motor1_dopredu(int pwm) {
  // zde dáte program pro točení motoru1 dopředu
  pwm = map(pwm, 0, 100, 150, 255);
  /*funkce map převede aktuální hodnotu pwm, na hodnotu 150-255.
Čísla 0, 100 říkají jakou hodnotu bude mít vstupní parametr pwm*/
  analogWrite(motor1_pwm, pwm);
  return pwm;  // vrátí přepočtenou hodnotu pwm
}
```

Na sériovém monitoru je vidět přepočet hodnoty 0-100% na hodnotu 150-255 pro pwm regulaci. Pozor hodnota 0% bude přepočtena na hodnotu 150. Proto je pro praktické použití dodat podmínku na 0%výkonu se rovná 0 pwm.

Knihovny

Knihovna je něco, jako soubor funkcí, které nám usnadňují programování Arduina. V podstatě využíváte něco, co už vám někdo předem naprogramoval. Arduino je Open Source platforma. To znamená, že i Arduino knihovny jsou na internetu zdarma. V knihovnách je zpracován kód pro ovládání jednotlivých hardwarových komponent, nebo kód pro různé softwarové komponenty či různé softwarové funkcionality apod.

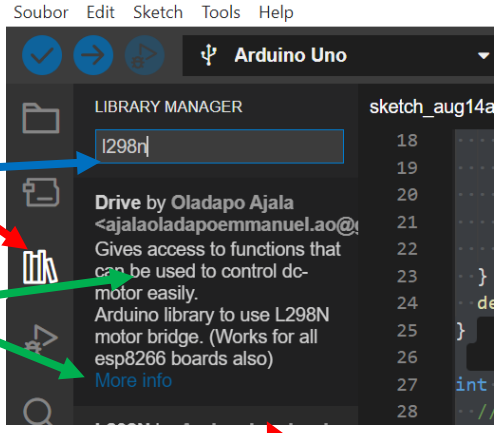
S instalací Arduino IDE je instalována sada standardních knihoven, které jsou ihned k dispozici. Existuje však nepřeberné množství knihoven od různých tvůrců, zpracovávající nejrůznější hardware a software, které nejsou součástí základní instalace. Získat je můžeš různými způsoby: stažením z internetových stránek tvůrce, nebo přímo ze seznamu v Arduino IDE. Nezapomeňte také, že po úspěšném importování knihovny máte většinou k dispozici příklady a ukázkové programy z dané knihovny. Příklady naleznete v kartě „File“ a záložce „Examples“.

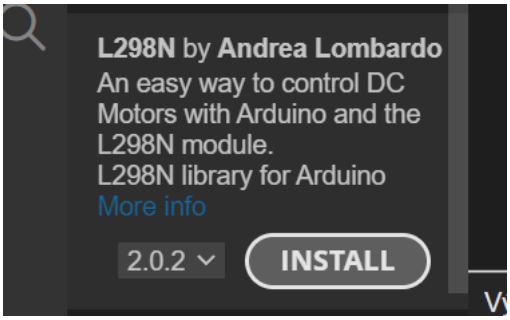
Instalace ze seznamu Arduino IDE

Jako nejjednodušší způsob instalace knihovny je možnost instalovat knihovnu přímo v Arduino IDE. Instalace je jednoduchá a velkou výhodou je, že Arduino IDE ti dá vědět, když vývojář zveřejní novou verzi knihovny a nabídne ti její aktualizaci. Výhody tohoto způsobu získání knihovny jsou tedy zřejmé – jednoduchá standardní instalace a jednoduché aktualizace. Avšak, aby byla potřebná knihovna k dispozici pro tento způsob instalace, je nutné, aby byl vývojář knihovny registrován na stránkách Arduina a mohl svoje knihovny distribuovat právě pomocí tohoto systému. Je však mnoho knihoven, které do tohoto systému zapojeny nejsou a distribuci si jejich vývojáři řeší jiným způsobem – např. pomocí systému GitHub.

Nyní naistalujeme knihovnu pro ovládání motorů. My jsme si vlastně tuto knihovnu již sami naprogramovali.

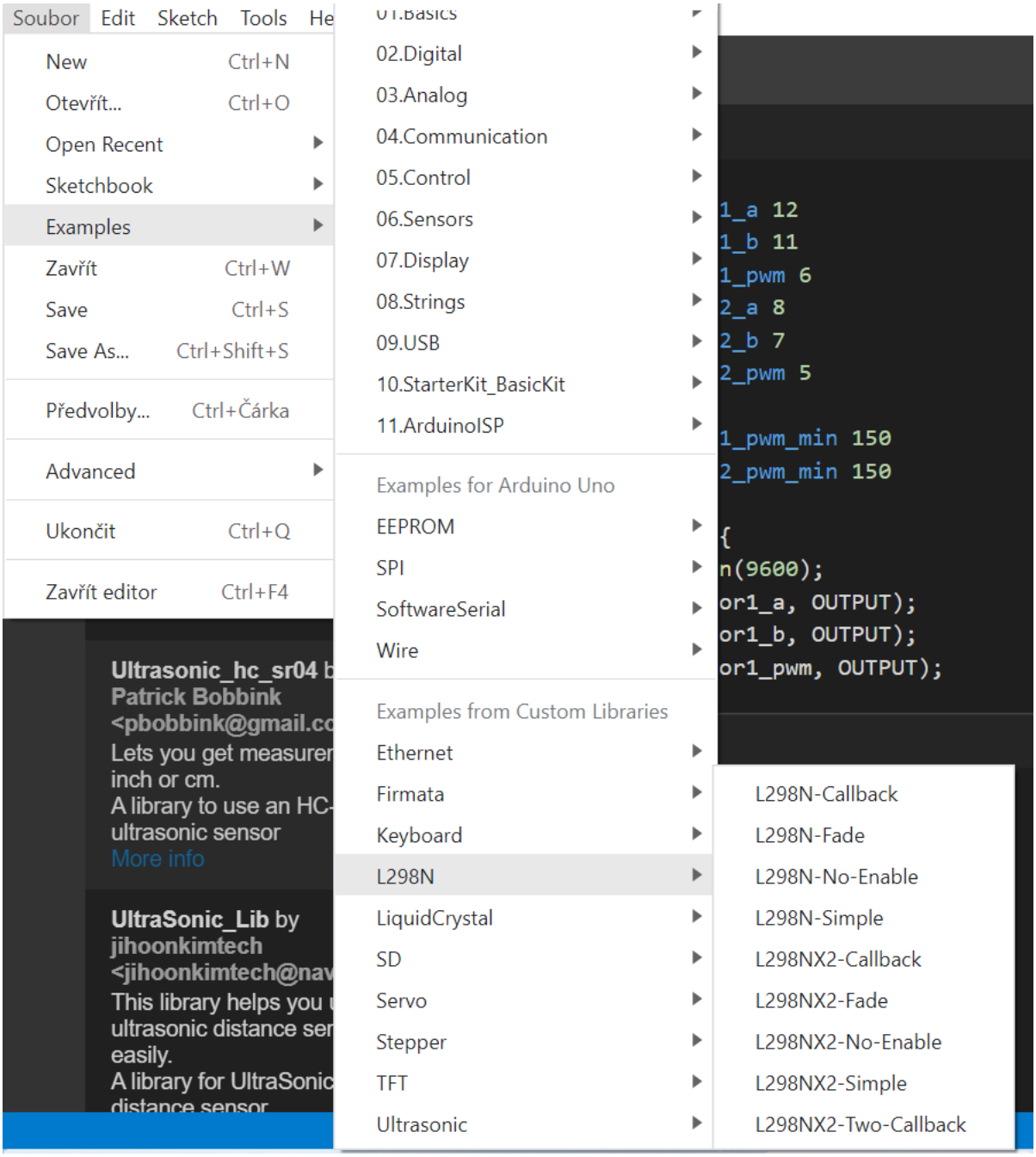
1. Klikneme na ikonu knihoven
2. Do vyhledávacího řádku napíšete co hledáte. V našem případě L298N a potvrdíte.
3. Seznam dostupných knihoven s popisem a většinou i s odkazem na podrobné informace na internetu





4. Najdi tuto knihovnu
5. Klikni a nainstaluj knihovnu

6. Většina knihoven obsahuje i ukázkové programy ty najdete tímto způsobem



Úkol č.20 – Odzkoušení práce s knihovnou

V hlavní smyčce volejte funkce v tomto pořadí

1. Motor1_dopředu – pwm 255
2. Čekej 3000ms
3. Motor1_stop
4. Čekej 3000ms
5. Motor1_dozadu – pwm 200
6. Čekej 3000ms
7. Motor1_stop
8. Čekej 3000ms

Motor1 by měl jet chvíli dopředu, zastavit, dozadu, zastavit a pořád dokola opakovat.

```
#include <L298N.h>

#define motor1_a 12
#define motor1_b 11
#define motor1_pwm 6
#define motor2_a 8
#define motor2_b 7
#define motor2_pwm 5

L298N motor(motor1_pwm, motor1_a, motor1_b);   //vytvoření instance pro
jeden motor

void setup() {
  // Used to display information
  Serial.begin(9600);
}

void loop() {
  motor.setSpeed(255);  // výkon motoru 0-255
  motor.forward();      //směr otáčení
  printSomeInfo();      //Informace o motoru + prodleva


  motor.stop();         // zastavení motoru
  printSomeInfo();      //Informace o motoru + prodleva

  motor.setSpeed(200);  // výkon motoru 0-255
  motor.backward();     //směr otáčení

  motor.stop();
  printSomeInfo();      //Informace o motoru + prodleva
}

//Print some informations in Serial Monitor
void printSomeInfo() {
  Serial.print("Motor is moving = ");
  Serial.print(motor.isMoving());
  Serial.print(" at speed = ");
  Serial.println(motor.getSpeed());
  delay(3000);  //čekej
}
```

Úkol č.21 – Odzkoušení práce s knihovnou pro dva motory

```
#include <L298NX2.h>  //pozor jiná knihovna

#define motor1_a 12
#define motor1_b 11
#define motor1_pwm 6
#define motor2_a 8
#define motor2_b 7
#define motor2_pwm 5

L298NX2 motor(motor1_pwm, motor1_a, motor1_b, motor2_pwm, motor2_a,
motor2_b);  //vytvoření instance pro dva motory

void setup() {
  // Used to display information
  Serial.begin(9600);
}

void loop() {
  motor.setSpeed(255);  // výkon motoru 0-255
  motor.forward();      //směr otáčení oba motory
  printSomeInfo();      //Informace o motoru + prodleva


  motor.stop();     // zastavení obou motorů
  printSomeInfo();  //Informace o motoru + prodleva

  motor.setSpeedA(255);  // výkon motoru A 0-255
  motor.setSpeedB(200);  // výkon motoru B 0-255
  motor.forward();       //směr otáčení obou motorů
  printSomeInfo();       //Informace o motoru + prodleva

  motor.stop();
  printSomeInfo();  //Informace o motoru + prodleva

  motor.setSpeedA(200);  // výkon motoru A 0-255
  motor.setSpeedB(255);  // výkon motoru B 0-255
  motor.forwardA();      //směr otáčení motoru A
  motor.backwardB();     //směr otáčení motoru B
  printSomeInfo();       //Informace o motoru + prodleva

  motor.stop();
  printSomeInfo();  //Informace o motoru + prodleva
}

//Print some informations in Serial Monitor
void printSomeInfo() {
  Serial.print("Motor A is moving = ");
  Serial.print(motor.isMovingA() ? "YES" : "NO");
  Serial.print(" at speed = ");
  Serial.println(motor.getSpeedA());
  Serial.print("Motor B is moving = ");
  Serial.print(motor.isMovingB() ? "YES" : "NO");
  Serial.print(" at speed = ");
  Serial.println(motor.getSpeedB());
  Serial.println();
  delay(3000);  //čekej
}
```

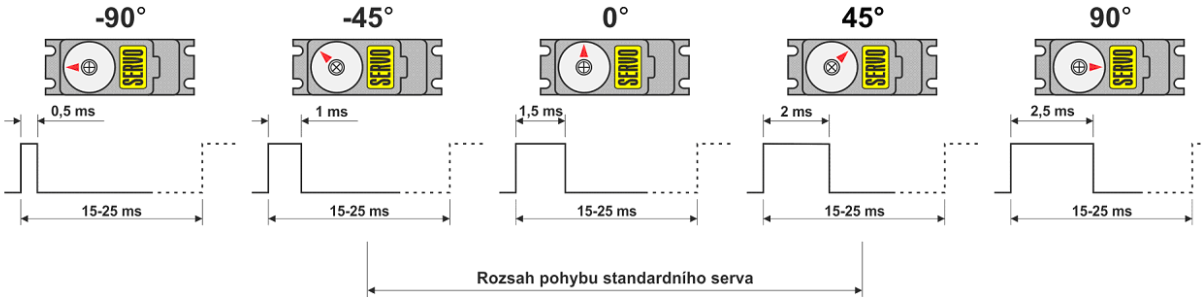
V hlavní smyčce volejte funkce v tomto pořadí

1. Motor1+2 dopředu – pwm 255
2. Čekej 3000ms
3. Motor1+2 stop
4. Čekej 3000ms
5. Motor1_dopředu – pwm 255 Motor2_dopředu – pwm 200
6. Čekej 3000ms
7. Motor1_doradu – pwm 200 Motor2_dopředu – pwm 255
8. Čekej 3000ms

Servo

Modelářské servo je elektromotor s převodovkou doplněný elektronikou, která zajišťuje řízení elektromotoru s využitím zpětné vazby od polohy výstupního hřídele (tzv. polohový servopohon). Požadovaná poloha výstupního hřídele je určena vstupním signálem, kde doba trvání impulzu odpovídá úhlu natočení.

Běžná serva poskytují rozsah pohybu cca 0°-180° (šířka impulzu 500 - 2500 ms). Střední poloha odpovídá impulzu o šířce 1500 ms, opakování impulzů je s frekvencí cca 50 Hz.



V programu nebudeme jezdit úplně z kraje do kraje (výchylka 0 – 180°), ale omezíme se na ±60° (tj. 30 až 150º), to určitě zvládne každé servo. Stejně každý typ serva má trochu jiné výchylky a málokdy zadávaný úhel „sedí“.

Pro práci se servo motorem poskytuje Arduino třídu „Servo“. Aby bylo možné ji použít je potřeba zahrnout do programu její knihovnu příkazem *#include <Servo.h>*. Následně se pro každý ovládaný motor vytvoří instance této třídy *Servo myservo;*. Aby bylo možno s každým motorem pracovat samostatně, je potřeba jej připojit na samostatný pin, v našem příkladu je to pin D9. To, na který pin má Arduino posílat informace, se mu řekne pomocí *myservo.attach(9);*. A konečně pro natočení motoru na určitý úhel stačí použít *myservo.write(pos);*. Tím se motor natočí na úhel, který aktuálně obsahuje proměnná *pos*.

Úkol č.30 – Pohyb serva

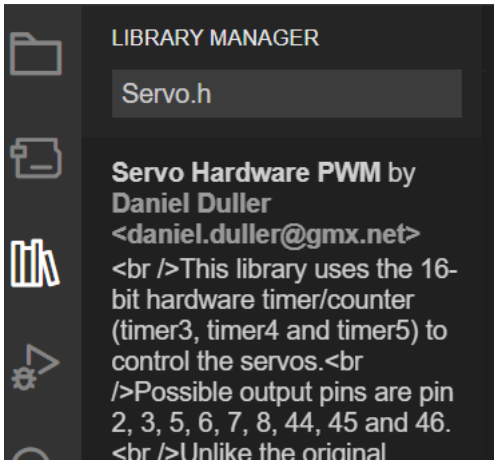
Odzkoušejte jak v simulátoru, potom na autíčku

```
#include <Servo.h>    //zahrnutí knihovny pro ovládání servo motoru
Servo myservo;        //každý motor má svou instanci třídy Servo
int pos ;             //proměnná obsahující pozici motoru (úhel natočení)

void setup()
{
  myservo.attach(9);    //tento motor je připojen na pin D9
}

void loop()
{
  for(pos = 30; pos <= 150; pos ++) //je od úhlu 30° do úhlu 150°
  {
    myservo.write(pos);  //natočení motoru na aktuální úhel
    delay(20);           //chvilka čekání než se motor natočí
  }
  for(pos = 150; pos >= 30; pos --) //je od úhlu 150° zpět do úhlu 30°
  {
    myservo.write(pos);  //natočení motoru na aktuální úhel
    delay(20);           //chvilka čekání než se motor natočí
  }
}
```

Úkol č.31 – Pohyb serva 6x

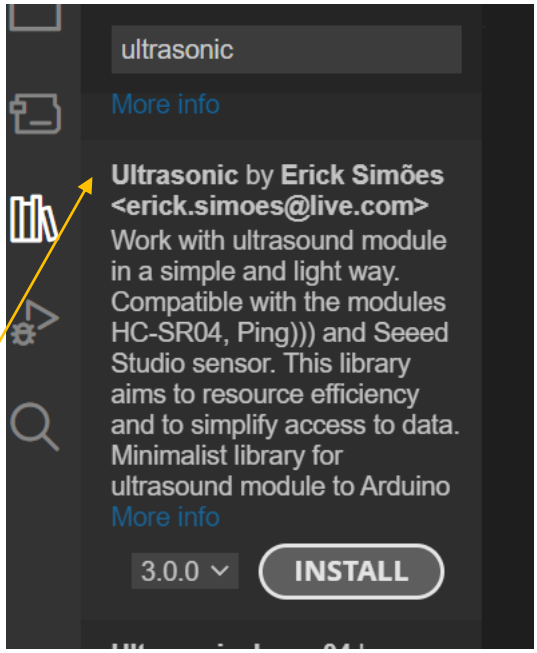
Upravte program pro ovládání 6-ti serv. 3 serva na začátku jdou od úhlu 50° do 130° zbylá tři serva jdou opačně (130° - 50°). Každý motor musímít svou instanci třídy (objekt) Servo (jedinečný název myservo1, myservo2 …). Každému motoru je zapotřebí přiřadit ovládací pin (attach()) a potom je řídit pomocí příkazu write. Odzkoušejte na simulátoru.

Pokud budete potřebovat přesnější a nezávislé na programovém vybavení je možno využít řízení serva pomocí pwm. Potom využijte knihovnu. Pro její využití je zapotřebí si nastudovat popis knihovny a její použití. To nebudeme nyní zkoušet.

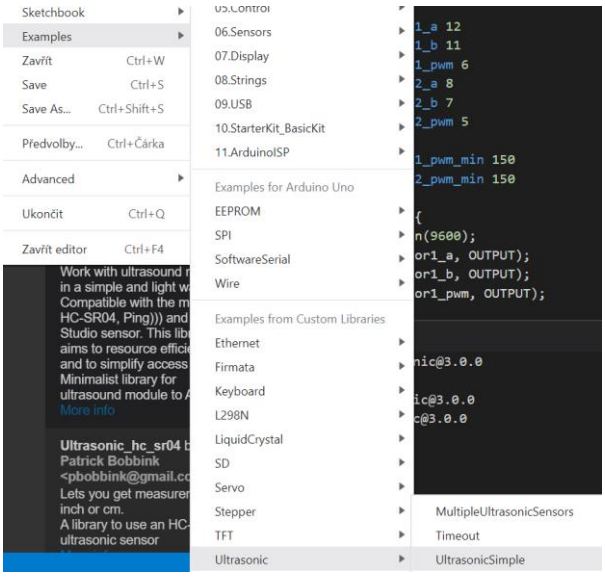


Ultrazvukový měřič vzdálenosti HC-SR04

Automatické měření vzdálenosti nemusí být zdaleka takový problém, jak by se mohlo na první pohled zdát. Pomocí ultrazvukového principu můžeme vzdálenost měřit velmi pohodlně, bezkontaktně a dokonce přesně. Typickým příkladem je použití měřiče jako detektor překážky pro Arduino robota. Pomocí Arduina aktivujeme signál "TRIG" na měřiči vzdálenosti, a to na dobu minimálně 10 µs. Po této aktivaci vyšle modul ultrazvukový signál na cca 40KHz a čeká na jeho odražení od překážky a zpětné zachycení měřičem (přijímačem). Mezi tím se aktivuje signál "ECHO", jehož délka je pak úměrná vzdálenosti překážky. Signál "ECHO" se deaktivuje po zachycení odraženého signálu měřičem vzdálenosti. Pozor pro přesnost měření je zapotřebí po skončení měření počkat nějakou dobu, než se ztratí veškerý odražený signál. Čím je měřená vzdálenost větší je potřeba počítat i s delším časem pro měření. V době měření procesor musí čekat na odražený zvuk a nemůže zpracovávat hlavní program. Existuje mnoho knihoven které umí obsloužit HC-SR04, některé výsledek měření odesílají v palcích jiné měří na mm nebo cm. Některé knihovny měří s takovou přesností, že je zapotřebí udat teplotu okolí. My si vybereme knihovnu, která je jednoduchá na ovládání.



Nainstalujte knihovnu Ultrasonic by Erick Simões



Vyberte ukázkový program UltrasonicSimple

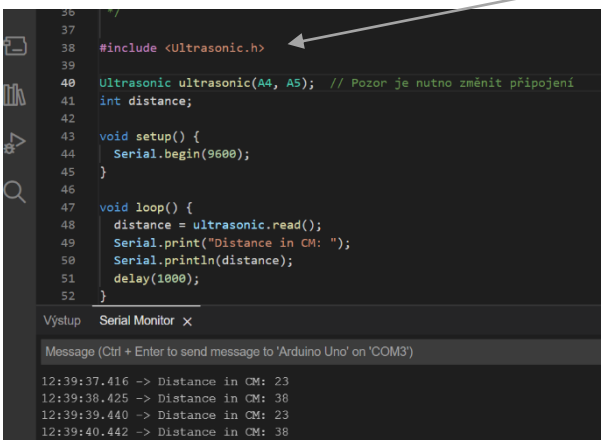
Soubor

Examples

Ultrasonic

UltrasonicSimple

Překladač připojí knihovnu Ultrasonic.h k našemu programu. Soubory s koncovkou h jsou tzv. hlavičkové soubory. Tam najdete definice a volání programu s koncovkou cpp.



Zde inicializujeme měřič vzdálenosti. Je nutno ji provést ještě před části setup. Nastavujeme piny kam je připojen. V našem případě je nutno tento řádek upravit z důvodu jiného připojení:

Ultrasonic ultrasonic(A4, A5);

ultrasonic.read() přečte vzdálenost a uloží do proměnné pro další použití

Časové zpoždění před další měřením

Výsledky měření jsou zaslány na sériový port

Úkol č.40 – Měření vzdálenost

Použijte předešlý program a zkuste přesnost měření vzdálenosti. Výsledek měření odešlete po sériové lince do PC. Proveďte minimálně 10 měření z Arduina. Pokud je více než 3 měření o 10% odlišných od průměrného měření považujte měření arduina za nemožné: - Měření provádějte ve dvojicích a tabulku překreslete a vypište do sešitu. Podle okolního rušení je přesnost měření. Pokud program nenajde překážku tak je možné, že výsledek měření bude menší než 10.

	Naměřeno cm	Rozdíl údajů		Údaj z Arduina cm
		cm	%	
Vzdálenost předmětu cca 1 cm				
Vzdálenost předmětu cca 5 cm				
Vzdálenost předmětu cca 10 cm				
Vzdálenost předmětu cca 20 cm				
Vzdálenost předmětu cca 50 cm				
Vzdálenost předmětu cca 75 cm				
Vzdálenost předmětu cca 1 m				
Vzdálenost předmětu cca 2 m				
Nejmenší měřená vzdálenost				
Největší měřená vzdálenost				

Úkol č.41 – vzdálenost

Většina knihoven má možnost dalšího nastavení než je defaultní. Tato knihovna má možnost měnit maximální čas, který program čeká na příchod odraženého signálu. Proto do funkce setup doplňte řádek.

```
42
43  void setup() {
44    Serial.begin(9600);
45    ultrasonic.setTimeout(40000UL);
46  }
47
```

Změřte nyní nejdelší vzdálenost, kterou lze změřit pomocí senzoru.

Největší měřená vzdálenost s ultrasonic.setTimeout(40000UL) je : cm

Úkol č.42 – rychlost měření

Zkuste zmenšit konstantu v delay(), aby výsledek měření byl ještě použitelný pro řízení autíčka. Testujte na vzdálenost cca 15cm.

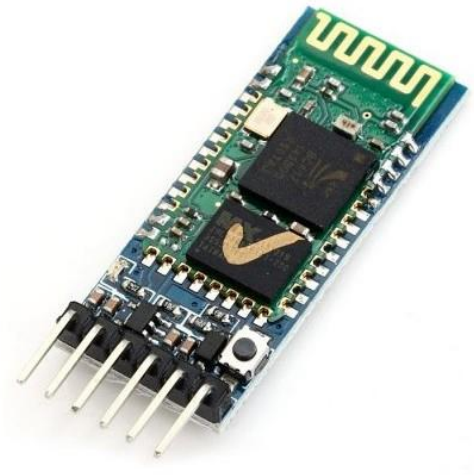
Úkol č.45 – Prohlédni si svoje okolí

Slučte program 30 (pohyb serva) a 40 (měření vzdálenosti) a pomocí sériového plotru si nechejte vykreslit okolí senzoru, když s ním budete otáčet pomocí serva. Doporučuji měřit vzdálenost pouze při pohybu na jednu stranu. Přiložte snímek obrazovky ze sériového plotru k programu.

Úkol č.46 – Auto 10cm

Napište program, který dojede rovně autíčkem k překážce a zastaví cca 10cm před ní. Využijte servo a měření vzdálenosti.

Bluetooth modul HC-05



S Arduinem již umíme komunikovat po kabelu, ale nebylo by lepší to umět i bez něj? K tomu použijeme bluetooth modul a budeme komunikovat přes něj. Arduino Bluetooth modul HC-05 je komunikační modul, který umožňuje propojit Arduino bezdrátově s dalším zařízením, které podporuje Bluetooth. Tento modul obsahuje Bluetooth ve verzi 2.0 a komunikuje s Arduinem pomocí sériové linky s výchozí rychlostí 9600 baudů. Vzhledem k velikosti celého modulu je i anténa poměrně malá a dosah je proto omezen na vzdálenost maximálně 10 metrů na volném prostranství. Co se týká napájecího napětí, tak pro tento modul HC-05 je zapotřebí napětí v rozsahu 3,3 až 6 Voltů. Proudový odběr při napájení 5 Volty se pohybuje okolo 2 miliAmpér v klidu a při komunikaci dosahuje maximálně 40 mA. Celý modul má pak rozměry 3,2 x 1,6 centimetrů. Pro úspěšné propojení Bluetooth modulu HC-05 s Arduinem stačí zapojit celkem čtyři vodiče. Propojíme RXD (podle druhu modulu je zapotřebí použít odporový dělič pro 3,3V logiku na vstupu), TXD, GND se zemí Arduina a VCC s 3,3 nebo 5 Volty Arduina. Nejjednodušší je zapojení modulu na hardwarový RS232 (sériový port). Ten se nachází na pinu D0 a D1 a můžete hned komunikovat. Nevýhoda je v tom, že tento hardwarový sériový port je využíván pro programování arduina a není možné připojit dvě aktivní zařízení. Další hardwarový sériový port arduino UNO nemá, proto je zapotřebí si pomocí softvérového sériového portu.

Úkol č.31 – Auto 10cm

Napište program, který dojede rovně autíčkem k překážce a zastaví cca 10cm před ní. Využijte možnost řízení rychlosti jízdy. Před překážkou pozvolně zpomalujte.

Úkol č.32 – Auto

Napište program pro jízdu auta okolo překážky.

1. Narovnejte senzor dopředu auta.
2. Jeďte dopředu autem rovně k překážce
3. Zastavte cca 8cm od překážky
4. Otočte senzor na úhel 0°
5. Otáčejte autem na místě, dokud nebude vzdálenost od překážky 12cm
6. Jeďte okolo překážky a udržujte vzdálenost od překážky cca 12cm